AD-A119 801 STANFORD UNIV CA PROJECT ORION F/G 9/2
HARDWARE FOR KINEMATIC STATISTICAL GRAPHICS.(U)
NOV 81 J H FRIEDMAN, W STUETZLE N00014-81-K-0340
UNCLASSIFIED ORION-005 NL

1 OF 1
AD A
119801

END
DATE
FILMED
~~11-82~~
DTIC

2

HARDWARE FOR KINEMATIC STATISTICAL GRAPHICS

Jerome H. Friedman

Werner Stuetzle

ORION 005

NOVEMBER 1981

Project ORION

Department of Statistics
Stanford University
Stanford, California

4

HARDWARE FOR KINEMATIC STATISTICAL GRAPHICS

by

Jerome H. Friedman

and

Werner Stuetzle

Stanford University
and
Stanford Linear Accelerator Center

ORION 005

November 1981

PROJECT ORION

Department of Statistics
Stanford University
Stanford, California

HARDWARE FOR KINEMATIC STATISTICAL GRAPHICS

Jerome H. Friedman
Stanford Linear Accelerator Center
Stanford University

Werner Stuetzle
Stanford Linear Accelerator Center
and
Department of Statistics
Stanford University

ABSTRACT

The hardware requirements for a computer graphics system capable of supporting kinematic statistical graphics are specified. The various options are discussed, and the **Orion–1** workstation currently in use at the Stanford Linear Accelerator Center is described.

This work was supported by the Department of Energy under contracts DE–AC03–76SF00515 and DE–AT03–81–ER108 43 and by the Office of Naval Research under contract ONR N00014–81–K–0340.

1. Introduction

When the pioneering **Prim–9** system for kinematic display of high-dimensional statistical data (Fisherkeller, Friedman and Tukey, 1974) was conceived and implemented at the Stanford Linear Accelerator Center (SLAC), it required specialized and expensive communications and graphics hardware and a non negligible part of the computing power of a IBM 360/91. This was probably the reason why the system, despite receiving widespread attention, saw little practical use and did not become a generally accepted tool for data analysis. Since then the price of computers and graphics equipment has dropped drastically, so that a hardware configuration supporting kinematic graphics can now be built for about $25000. In view of these developments the **Orion–1** workstation for kinematic graphics was designed and built at SLAC. It is based on new and inexpensive microprocessor and raster graphics technology.

In this report we specify the requirements on the hardware of a computer system capabable of supporting kinematic graphics of the type used in **Prim–9**. We describe the **Orion–1** workstation now in use at SLAC, and we motivate our design decisions.

2. Prim–9: A brief description

We will give a brief description of **Prim–9**, mainly for the sake of completeness and because the original publication is not easily accessible. No written account can do justice to the system; to appreciate its qualities one has to see the film produced at SLAC and available upon request (write to Computation Research Group, Stanford Linear Accelerator Center, Stanford, Ca. 94305).

Prim–9 was a system for visual inspection of up to nine dimensional data, mainly intended for the detection of clusters and hypersurfaces. The computer system on which it was implemented allowed the modification of pictures in real time and thus made it possible to generate movie-like effects. **Prim** is an acronym for the basic operations of the system:
Projection: The observations could be projected on a subspace spanned by any pair of the coordinates. The resulting scatterplot was shown on a display screen.
Rotation: A subspace spanned by any pair of coordinates could be rotated. If the projection subspace and the rotation subspace shared a common coordinate, the rotational motion caused the user to perceive a spatial picture of the data as projected onto the three dimensional subspace defined by

the coordinates involved. When the user terminated rotation in a particular plane, the old coordinates in that plane were replaced by the current (updated) coordinates. This made it possible to look at completely arbitary projections of the data, not necessarily tied to the original coordinates.
Masking: Subregions of the observation space could be specified, and only points inside the subregion were displayed. Under rotation, points would enter and leave the masked region.
Isolation: Points that were masked out (i.e. not visible) could be removed, thus splitting the data into two subsets.

We will call any system for kinematic display of high dimensional data using these or similar basic operations a **Prim** system.

3. Hardware Requirements

Hardware supporting **Prim** must have the ability to compute and draw new images at a rate of at least five times per second. Such a low update rate is aesthetically not very pleasing, but it can be sufficient if the controls, the means by which the user interacts with the program, are suitably designed. Operations that have to be performed for every update of the image are called real time operations.

The basic real time computation for **Prim** is rotation of a set of points. Let $\mathbf{x}_1 \ldots \mathbf{x}_n$ denote a sample of points in $\mathbf{R}^3$, (generally obtained by projecting higher dimensional observations on a 3-dimensional subspace) and let $\mathbf{R}$ denote a $3 * 3$ rotation matrix. The user controls the rotation matrix, and the rotated points $\mathbf{R}\mathbf{x}_1 \ldots \mathbf{R}\mathbf{x}_n$ have to be computed for each new image displayed on the screen. In the simplest case only two coordinates of the rotated points (corresponding to the horizontal and vertical screen axis) are needed. It is sometimes desirable to use the 3rd coordinate for depth cueing, or to work with $4 * 4$ rotation matrices in homogeneous coordinates to draw perspective images of the point cloud, but we will ignore this here. Computing two coordinates of $\mathbf{R}\mathbf{x}_1 \ldots \mathbf{R}\mathbf{x}_n$ requires $6n$ multiplications, $4n$ additions plus operations to access and store the matrix elements and the coordinates. One would probably write the rotation routine in assembly code, and the total number of operations will vary significantly from processor to processor, depending on the number of registers, available address modes, etc. In many cases, for example if the rotations are performed on a microproceesor, the time critical operation is the multiplication, and rotating 1000 points 10 times a second requires a multiplication time of substantially less than 10 microseconds, when the overhead and the time necessary to draw the images is taken into account.

A **Prim** system for which rotation is the only real time computation can certainly be useful, but it is highly desirable to have additional capabilities, for example real time scatterplot smoothing (Cleveland, 1979; Friedman and Stuetzle, 1982), computation of near-neighbor density estimates (Tukey and Tukey, 1982) computation of classification rules, etc.

The basic requirement for the display part of a **Prim** system is the ability to draw scatterplots in real time. Assume that we want to draw a scatterplot of 1000 points 10 times a second. If the coordinates of each point on the screen are specified by two 16 bit integers, this requires a data transmission rate of more than 300 K bits per second (300 Kbaud).

An important choice is between color display and monochrome display. The availability of color opens up a whole new range of possibilities. It is much more efficient for distinguishing groups than blinking, highlighting, or use of special symbols, and it allows one to represent a fourth dimension on the screen. We have found color to be highly desirable. Also highly desirable is the ability to draw vectors in real time, for example to draw the minimal spanning tree of the observations or to draw a smooth through a scatterplot. As a minimal spanning tree has $n - 1$ edges, drawing it for 1000 points 10 times per second requires the ability to put up 10000 vectors per second.

Ability to draw not only scatterplots and vectors, but also color maps, in real time would be very useful. Color maps could for example be used *to represent two-dimensional marginal densities.* For technical reasons to be discussed in section 5 only very little energy has been spent so far on techniques requiring real time drawing of color maps.

The purpose of the control part of the **Prim** hardware is the translation of a motion of the hand into a motion on the screen. Almost any input device can be used for control, if it is suitably used by the program and if its effect is adapted to the available update rate. We will further discuss this point in section 6.

4. Computer Hardware

A basic choice that has to be made is between implementing **Prim** on a multi user mainframe connected to a graphics terminal (terminal concept) or implementing it on a workstation (workstation concept). A workstation by definition performs the real time computations for **Prim** on its own dedicated processor, whereas a terminal has very little local computing capability and thus relies heavily on the mainframe it is connected too. Up till now all **Prim** systems were implemented on mainframes. (The Evans and Sutherland Picture System 2, which was used in two of the systems, has a built in matrix multiplier and thus can rotate points, but it is not a general purpose processor). The adherence to the terminal concept was dictated by technical necessity; there were no inexpensive processors available, on which a workstation could be based. Unfortunately, as noted above, the graphics terminal has to be connected to the mainframe with a line allowing for a transmission rate of several hundred Kbaud, and ports of that speed are highly non standard and are generally not supported by mainframe manufacturers. The terminal concept poses some additional problems. The update rate will depend on the current load on the mainframe, and the remedy of giving the graphics job a high priority tends to aggravate other users. If there is a charge for CPU time, **Prim** will be expensive to run, because kinematic graphics, as noted above, is very CPU intensive. The conclusion thus seems to be that **Prim** should be implemented on a graphics workstation. The workstation can have various degrees of self sufficiency. At the one extreme its computer part can consist of a single CPU board, relying on a mainframe for editing, storage and compilation of programs and for preparation of data, and being connected to the mainframe by a standard serial communication line used to download programs and data in binary form. At the other extreme it can be a self sufficient personal computer, with its own operating system, disks, editors, compilers, etc. We will further discuss the pro's and con's of the two approaches below.

The next question is on what kind of processor the workstation should be based. The eight bit microprocessors that have been commercially available for some time are the first possibility that comes to mind. However, they generally do not even have hardware integer multiplication, and their small address range and low overall speed make them unattractive for development work starting now, especially in view of the availability of 16 bit microprocessors like the Zilog Z-8000 and the Motorola MC-68000, which are about 10 times as powerful as for example the Z-80. The MC-68000 running at a 8 Mhz clock rate can perform a 16 bit integer multiplication in about 8 microseconds. It thus has enough computing power for the basic real time computation of **Prim**, rotation of a set of points.

In contrast to CPU boards, personal computers based on the MC-68000 are just now becoming available. Therefore there is very little experience with them. Implementing **Prim** on a personal computer based on the MC-68000 seems very attractive. There are, however, two possible pitfalls:

—The operating system and utilities of these personal computers are not written with real-time applications in mind. If, due to certain idiosyncrasies in the system, these applications run too slowly, this might not

be easy to repair, because the manufacturers generally are very reluctant to distribute circuit diagrams and source code of operating system and utilities.
—If the display is not integrated, it has to be interfaced, which is more difficult for a personal computer than for a single board processor, in particular when documentation, as usual, is rudimentary or non existent.

Real time computation of smooths, density estimates, classification rules etc requires a processor with fast real arithmetic. Floating-point chips like the AMD 9511 are much to slow. Commercially available array processors are unsuitable for the following reasons:
—They are very expensive ($50000 +).
—They usually are not easy to program, and the utilities that come with them are geared toward applications in digital signal processing.
—If they are programmable at all, the program memory is insufficient.

At the moment there seems to be no commercially available processor that could serve as basis for a workstation and would be fast enough to allow advanced real time computations.

5. Display Hardware

There are two kinds of displays suitable for real time applications, vector displays and raster displays.

A vector display basically consists of a graphics processor, beam deflection electronics and a CRT tube. The program for the graphics processor is called the display file; it is usually stored in the display's private memory in binary form. The graphics processor understands instructions like "move beam to screen location (x,y)","draw vector from location (x_1, y_1) to (x_2, y_2)","jump to location i in the display file", and it drives the beam deflection electronics accordingly. The display file is executed periodically, to refresh the image on the CRT.

A raster display is based on the same principle as a television set; the three electron beams (resp the single beam in a monochrome display) scan the entire screen a fixed number of times per second. The screen is divided into little dots, called pixels, 640 * 480 of them in standard TV format. Associated with each pixel are one or more bits of display memory, containing the information on the color and intensity of the pixel. As the electron beams scan the screen, electronic curcuits read the display memory and modulate the beams accordingly.

Vector displays have the advantage of high precision: They have up to 4096 * 4096 addressable screen locations. They can draw very complicated images quickly; the Evans and Sutherland Picture System 2, for example, permits updating of an image with 10000 vectors or points 20 times per second.

On the other hand vector displays cannot draw areas, only points and vectors. Most of them are monochrome, and if they have color, it is of poor quality. Furthermore they are expensive.

Raster displays usually allow for high quality color. They can draw areas, not just vectors and points. Raster displays are less expensive than vector displays. A large fraction of the costs of a raster display goes into the display memory. (A system with 640 * 480 pixels and 8 bits per pixel for color and intensity information requires about 300 K bytes of display memory.) As memory prices continue to drop, the price of raster displays can be expected to drop accordingly. Personal computers with integrated raster display will then start to appear on the market.

On the other hand raster displays generally have lower resolution than vector displays, with 512 * 512 or 640 * 480 being common and 1024 * 1024 being the maximum. It is not possible to update vectors very quickly. Note that if a vector is given by its beginning and end coordinates, it has to be determined which pixels the vector crosses, and the corresponding display memory locations have to be changed. Although most raster displays have built in hardware to do that, drawing a vector can take anywhere from 0.2 to 80 microseconds per pixel. If we assume two microseconds per pixel for the moment and an average vector length of, say, 50 pixels, drawing 1000 vectors takes 1/10 th of a second, and thus an image with 1000 vectors can be updated at most ten times per second. (This estimate is highly optimistic, totally neglecting overhead and the time needed to erase the previous picture.)

We can conclude that raster displays can statisfy the basic requirements of **Prim**, and that they are the preferrable output device. There is a general tendency towards use of raster displays, with vector displays becoming more and more restricted to very specialized applications, for example computer aided design. The only tempting advantage of vector displays is the ability to draw pictures with many vectors; the high resolution is not crucial.

For the commonly used raster displays it takes several seconds to rewrite the entire display memory. Such displays thus do not allow real time drawing of color maps. Suitable systems are available, but their costs are still in the million dollar range.

6. Characteristics of a Raster Display

We will now list the most important Characteristics, that have to be considered in the choice of a raster display:

Resolution: The commonly available resolutions are 512*512, 640*480 and 1024 * 1024. For direct viewing the lower resolutions are sufficient. High resolution results in crisper characters, less staircasing in vectors, and might be advantageous if the screen is filmed with a movie camera. The larger display memory offers the possibility to, for example, store four subpictures, each still with 512 * 512 resolution, and to view them together or separately, using Pan and Zoom (see below). On the other hand high resolution systems are much more expensive. They require four times as much display memory, and the monitor is about four times as expensive as a low resolution monitor. The drawing of vectors takes longer, because a vector of the same length crosses twice as many pixels (see below). Even drawing points takes longer because, in order to be clearly visible, points have to consist of several pixels, and more pixels are needed in a high resolution system than in a low resolution system. Note however, that a high resolution system with Pan and Zoom can always be used like a low resolution system.

Refresh rate: Refresh can either be interlaced or non interlaced. Interlaced refresh means that the entire screen is refreshed every 1/30th of a second; in the first half of that time the even lines are refreshed, in the second half the odd lines. In non interlaced refresh the entire screen is written every 1/60th of a second. Non interlaced refresh is preferable. It allows the use of a monitor with short persistence phosphor without incurring flicker, thus reducing streaking on the screen, and it is particularly advantageous if one plans to take movies off the screen with a regular movie camera. On the other hand non interlaced refresh requires an expensive high bandwidth monitor, and there is at the time no inexpensive system that offers 1024*1024 resolution with non interlaced refresh.

Number of color planes: The number of color planes (bits of display memory per pixel) determines, how many colors can be shown on the screen simultaneously. Four planes, allowing for 16 colors, seem to be the minimum, eight planes are desirable. If eight planes are available, they can be divided into two sets of four planes each, and the image can be double buffered: while the image in one set of planes is currently displayed, the new image is written into the other set of planes. This is advantageous because it makes it possible to erase planes instead of selectively overwriting the previous points and vectors with the background color. Plane erase saves time, but if it is not used in conjunction with double buffering, it results in disturbing interference effects between update and refresh.

Lookup tables: A distinction has to be made between pseudo color displays and full color displays. In a pseudo color display the memory planes are directly assigned to the colors. In a three plane system, for example, plane 1 could control the red gun, plane 2 the green gun, and plane 3 the blue gun. This system would allow to show 8 different colors, but which colors those are would be determined once and for all, depending on the intensity settings of the guns. In a full color system, the bit pattern in the display memory location corresponding to a given pixel is interpreted as an address defining a location in each of three lookup tables, one for red, one for blue, and one for green. The value stored at that location in the red lookup table determines the intensity of the red gun, the value at the same location of the blue lookup table determines the intensity of the blue gun, etc. If each word in the lookup tables is six bits long, the system can dis v 2^{18} colors. How many of those can be shown simultaneously depends ne number of memory planes. If there are four planes, each lookup table h e be 16 words long, and for fixed setting of the lookup tables one can ch e between 16 out of the total of 2^{18} colors. Displays also vary in the length of the lookup table and, accordingly, in the precision of the dig - to-analog converters, that control the gun intensities. Commonly used word lengths are six bits and eight bits. A word length of six bits seems to be enough for all practical purposes. Full color is a necessity, if color is to be used for displaying a fourth dimension on the screen, or if it is used as a depth cue. Pseudo color is a severe restriction.

Hardware blink: It allows to make random pixels blink, depending on the value written into the corresponding display memory locations. In some systems a memory plane is permanently assigned to blink control and is not used in addressing the lookup tables, whereas in others this assignment can be done in software, thus avoiding the loss of a memory plane in situations where blink is not actually used. Hardware blink is not crucial in a color display, and it is certainly not important enough to waste a memory plane for it.

Pixel update speed: Pixel update speed is the speed at which information can be written into the display memory. One has to keep in mind, however, that an update speed of two microsecond per pixel does not mean, that a scatterplot of 1000 points can actually be drawn in 2 miliseconds. The time needed will to a large extent depend on the driver software, the speed of the interface, etc.

Vector drawing speed: Drawing vectors on a raster display is not entirely trivial; it is necessary to calculate, from the coordinates of the endpoints of the vector, which pixels it crosses. The speed, at which a vector can be written into display memory, is usually given in microseconds per pixel. The simplest displays rely on their host computer for this task. They are unsuitable, if one intends to draw a sizable number of vectors in real time. The same applies to displays that use a standard eight bit microprocessor like the Z-80 for vector drawing. They usually take between 50 and 100 microsecond per pixel. Displays that can be used to draw pictures with a moderate number (500) vectors in real time either contain a custom designed microprocessor or hard wired logic for vector drawing. Their speeds ranges from 0.1 to 4 microseconds per pixel.

Local intelligence: Some of the more recent raster displays contain high speed processors allowing local operations other than vector drawing. Examples for additional capabilities are drawing of multiple pixel points and multiple width vectors, area fills, and Newman and Sproull' s raster op (see Newman and Sproull, 1979, chapter 18). These extra capabilities are not crucial, but desirable.

Connection to the host: For use with **Prim** the possibility of connecting the display to the host via a parallel port is a necessity.

Hardware cursor: Displays with hardware cursor put up a cross or other symbol on the screen at a position controlled by an input device like trackerball or joystick, without help of the host. The host can read the cursor position, but it does not have to bother with drawing the cursor. A hardware cursor is highly desirable.

Pan and zoom: Pan and Zoom allow to choose part of the image and magnify it, so that it fills up the entire screen. This does not bring a gain in resolution; a linear magnification of 2 is achieved by displaying the information for each pixel in the selected region in a square of 4 pixels on the screen, a magnification of 3 by blowing up each pixel into a square of 9 pixels, etc. For a low resolution display Pan and Zoom do not seem to be worthwhile. If the display has high resolution, Pan and Zoom can be useful, because they allow for example the simultaneous or separate viewing of four subpictures, each still with 512 * 512 resolution.

External synchronization: It allows the supply of the vertical synch signal from an external source instead of an internal generator. We have not yet found an example where this can be useful.

RGB to NTSC converter: The video output generated by most raster displays is not the same as the video signal used in color television; the signals for the red, blue and green gun are brought out separately, whereas in color television they are encoded onto a single carrier. A RGB to NTSC converter takes as its input the red, blue and green signals and produces an output that can be shown on a standard television set or recorded on a video recorder. Such a converter can only be used for 640 * 480 resolution with interlaced refresh. Conversion results in a substantial loss of quality. (The bandwidth of a commercial TV set or video recorder is about 2.5 Mhz, whereas a usual RGB monitor has amplifiers with a bandwidth of 20 Mhz). It is not clear whether the combination of converter and video recorder can provide a recording of useful quality.

7. Controls

The design of the controls is a very important part of the overall design of **Prim**; it determines to a large extent how easy or difficult the system is to learn and to use. Input devices commonly associated with interactive graphics are graphics tablet, joystick, trackerball and buttons.

A graphics tablet is a rectangular pad, usually about the size of a legal pad, with a pen connected to it. The user moves the pen over the tablet. The display or host computer, to which the terminal is attached, can find out at any given time whether the pen is pressed down on the tablet, touches the tablet without being pressed down, or is away from the tablet, and in the first two cases it can read the coordinates of the contact point. It is probably not necessary to describe a joystick; joysticks are used as input devices in many of the recently popular video games. As the name indicates, a trackerball is a ball of about 2 to 3 inches in diameter, that is 2/3 embedded in a casing, with the remaining 1/3 sticking out on top. The ball can be spun in its casing by sliding the hand over it, and the position of the ball is translated into x and y coordinates. (The ball cannot be spun around the vertical axis of the casing.) Buttons or switches do not output a position; they only can be open or closed.

In **Prim** the input devices are mainly used for two tasks: controlling the rotation and positioning the cursor (for example to identify a point or to pick an item off a menu). These tasks can be performed with any one of the input devices, given suitable programming. We are thus discussing here only, which ones are the most convenient ones to use.

For controlling rotation the trackerball clearly is best suited, especially if the program is written in such a way that rotation of the ball translates into rotation of the point cloud in the natural manner. For cursor positioning and menu selection it has to be combined with buttons or switches: The user positions the cursor and then presses a button (enter-button) to tell the system that he is done. The main disadvantage of a trackerball is its price, generally around $1000.

A tablet is as nice for cursor positioning as a trackerball. An enter button is not needed, because the pen can be moved to the correct position and then pressed down; the micro switch in the pen acts as the enter button. A tablet is less natural than a trackerball for controlling rotation. However, it has other advantages. It can be useo easily to input curves and, combined with a character recognizer, it can even be used to enter numbers.

A joystick is fundamentally different from a trackerball and a tablet in that it is only suitable to control speed and direction of motion, not position. It is good for controlling rotation, but positioning of a cursor can be awkward.

Buttons or switches alone (not in combination with another input device) can be used for control. Indeed the original **Prim–9** system was entirely controlled by buttons. No choice was being made at that time; only buttons were available. Today a combination of trackerball and buttons seems to be the preferable choice, with joystick and buttons second. There is not enough experience with tablet control to warrant a judgement.

What was said above is only valid for moderate to high display update rates (more than, say, five updates per second). If the update rate is very low, the feedback between motion of the hand and motion on the screen gets lost. To prevent overshooting, the speed of motion, for example the angle by which the point cloud is rotated between one update and the next, has to be controlled by the program, leaving the user with the choice of direction. It is clear that in this situation the trackerball looses its advant age for rotation control.

8. The Orion–1 workstation at SLAC

The **Orion–1** workstation used at SLAC was designed in the summer of 1980 and was assembled between October 1980 and March 1981. This is important to keep in mind, because new hardware is coming on the market at a rapid rate, and a system designed today would certainly look different.

The workstation consists of a SUN-68000 microprocessor board, a 168E floating point processor, a Lexidata 3400 raster display, a trackerball and 6 switches for control, 16 bit parallel interfaces between the SUN board and the 168E and between the SUN board and the 3400, and a high speed serial link to the SLAC mainframe, an IBM 3081 (see diagram 1).

The SUN board is a MULTIBUS board developped by the Stanford

Computer Science Department for the Stanford University Network and now manufactured by several companies. (MULTIBUS is a trademark of INTEL corporation.) The board contains a 8 Mhz Motorola MC-68000 microprocessor, 256 K of RAM, two serial and one parallel port, a memory mapping and protection mechanism and a small monitor in ROM. The MC-68000 is the master processor for **Prim**, handling overall system control. One of the serial ports is connected to a terminal, the other one via a regular terminal line to the IBM 3081.

The 168E is basically a IBM 370 CPU without channels and interrupt capabilities. It was developed by SLAC engineers for the processing of particle physics data and has about half the speed of the genuine IBM 370 or about twice the speed of a VAX11/780. As it does not have any input/output facilities, it is used strictly as a slave processor. The MC-68000 loads program and data into the 168E memory via the MULTIBUS and then starts the 168E. The 168E executes its program and signals, when it is finished. The MC-68000 then reads the results of the computation out of the 168E memory, enters new input data, starts the 168E, etc.

The Lexidata 3400 has a resolution of 640 * 480 pixels and interlaced refresh. It is a full color display with 8 bit digital-to-analog converters. The pixel update speed is 2 microseconds per pixel, and a custom designed microprocessor allows vectors to be drawn at a speed of 3 microseconds per pixel. It has hardware blink, a hardware cursor and it comes with a 16 bit parallel port.

The high speed (100 Kbaud) serial interface between the MULTIBUS and the IBM 3081 emulates an IBM 3270 terminal. It was developed by the SLAC Computation Research Group.

Programs are edited and stored on the IBM 3081 mainframe. The MC-68000 is programmed mostly in Pascal, with time critical routines written in assembly language. The programs are translated into object code on the IBM 3081, using a cross compiler respectively a cross assembler, and are downloaded to the MC-68000 via the high speed serial link. The 168E is programmed in FORTRAN. The programs (which must not contain input/output statements) are translated by the IBM FORTRAN compiler, linked into an object module, translated into 168E microcode, and loaded into the 168E via the serial link.

The data to be analyzed are also stored on the IBM 3081, converted to binary representation and loaded through the serial link.

It is important to note that the high speed link is only used for downloading program and data into the 168E and the MC-68000. This could in principle be (and was until recently) accomplished through the regular RS-232 terminal connection between the IBM 3081 and the SUN board. The only difference is that this takes substantially longer. At execution time there is no interaction between the IBM 3081 and the **Orion-1** workstation; the workstation runs strictly stand-alone.

9. Discussion

As indicated in section 8, the **Orion-1** workstation at SLAC is not based on a personal computer. At the time of its design, inexpensive personal computers based on 16 bit microprocessors were not yet available.

The choice of a particular graphics system (Lexidata 3400) was based on considerations of the moment (price, rapid delivery, company record, etc). There are many companies making comparable equipment today. The decision for a low resolution display with interlaced refresh was due to financial constraints. Non interlaced refresh certainly would have been preferrable.

As far as the 168E is concerned, there was no choice to be made. The 168E was then and still is the only fast and inexpensive ($6000) numbercruncher in existence. Unfortunately it is not commercially available. It can be argued that it is not a good idea to base a tool for data analysis on components that are inaccessible to other users. Our consideration was that, first of all, the **Orion-1** workstation at SLAC is mainly a tool for research in statistical methodology, and it seemed unwise to limit the range of exploration because a particular piece of hardware is not yet generally available. This does not imply that an alternative system with more limited capabilities, based entirely on standard components, would not be useful. It could help to make **Prim** available to the community at large, and the reactions of others would provide valuable insight. Secondly about 75% of the code run on the **Orion-1** workstation resides on the MC-68000, with a clear separation of tasks between the 168E and the MC-68000 being enforced. The 168E could be replaced by a different floating point processor, as soon as a suitable one becomes available.

REFERENCES

Cleveland,W.S. (1979). "Robust locally weighted regression and smoothing scatterplots". *J. Amer. Statist. Assoc.* **74**, 829–836.

Fisherkeller,M.A., Friedman,J.H., Tukey,J.W. (1974). " PPRIM–9: An interactive multidimensional data display and analysis system". SLAC–PUB–1408.

Friedman,J.H. and Stuetzle,W. (1981). "Projection pursuit regression". *J. Amer. Statist. Assoc.* **76**, 817–823.

Newman,W.M. and Sproull,R.F. (1979). "Principles of interactive computer graphics". McGraw-Hill, Inc.

Tukey,P.A. and Tukey,J.W. (1982). "Graphic display of data sets in 3 and more dimensions". To appear in *Interpreting multivariate data,* V. Barnett, Editor. Wiley, London.

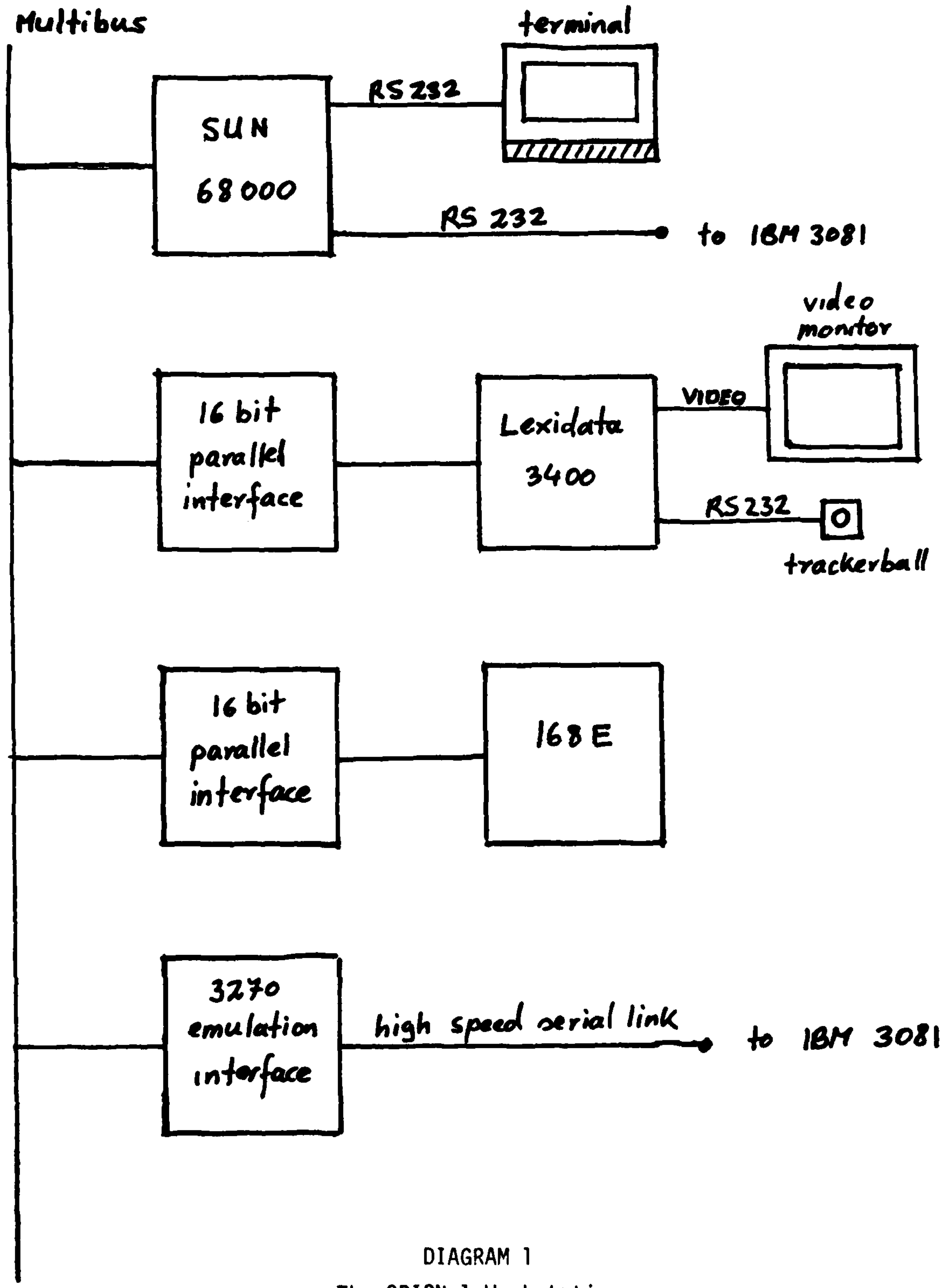

DIAGRAM 1

The ORION-1 Workstation

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ORION 005	2. GOVT ACCESSION NO. A119801	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) HARDWARE FOR KINEMATIC STATISTICAL GRAPHICS		5. TYPE OF REPORT & PERIOD COVERED Technical
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Jerome H. Friedman Werner Stuetzle		8. CONTRACT OR GRANT NUMBER(s) N00014-81-K-0340
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science Stanford University Stanford, California 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Department of the Navy Office of Naval Research Arlington, Virginia 22217		12. REPORT DATE November 1981
		13. NUMBER OF PAGES 15
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Navy position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The hardware requirements for a computer graphics system capable of supporting kinematic statistical graphics are specified. The various options are discussed, and the ORION-1 workstation currently in use at the Stanford Linear Accelerator Center is described.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)